PAuthKey: A Pervasive Authentication Protocol and Key Establishment Scheme for Wireless Sensor Networks in Distributed IoT Applications

Porambage, Pawani ; Schmitt, Corinna ; Kumar, Pardeep ; Gurtov, Andrei ; Ylianttila, Mika

Abstract: Wireless sensor Networks (WSNs) deployed in distributed Internet of Things (IoT) applications should be integrated into the Internet. According to the distributed architecture, sensor nodes measure data, process, exchange information, and perform collaboratively with other sensor nodes and end-users, which can be internal or external to the network. In order to maintain the trustworthy connectivity and the accessibility of distributed IoT, it is important to establish secure links for end-to-end communication with a strong pervasive authentication mechanism. However, due to the resource constraints and heterogeneous characteristics of the devices, traditional authentication and key management schemes are not effective for such applications. This paper proposes a pervasive lightweight authentication and keying mechanism for WSNs in distributed IoT applications, in which the sensor nodes can establish secured links with peer sensor nodes and end-users. The established authentication scheme PAuthKey is based on implicit certificates and it provides application level end-to-end security. A comprehensive description for the scenario based behavior of the protocol is presented. With the performance evaluation and the security analysis, it is justified that the proposed scheme is viable to deploy in the resource constrained WSNs.

DOI: https://doi.org/10.1155/2014/357430

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 357430, 14 pages
http://dx.doi.org/10.1155/2014/357430

Research Article

PAuthKey: A Pervasive Authentication Protocol and Key Establishment Scheme for Wireless Sensor Networks in Distributed IoT Applications

Pawani Porambage,[1] Corinna Schmitt,[2] Pardeep Kumar,[1] Andrei Gurtov,[3] and Mika Ylianttila[1]

[1] *Centre for Wireless Communications, University of Oulu, P.O. Box 4500, 90014 Oulu, Finland*
[2] *Department of Informatics, University of Zurich, Binzmühlestrasse 14, 8050 Zurich, Switzerland*
[3] *Department of Computer Science and Engineering, Aalto University, 00076 Aalto, Finland*

Correspondence should be addressed to Pawani Porambage; pporamba@ee.oulu.fi

Received 6 November 2013; Revised 2 May 2014; Accepted 7 May 2014; Published 13 July 2014

Academic Editor: Ken Choi

Wireless sensor Networks (WSNs) deployed in distributed Internet of Things (IoT) applications should be integrated into the Internet. According to the distributed architecture, sensor nodes measure data, process, exchange information, and perform collaboratively with other sensor nodes and end-users, which can be internal or external to the network. In order to maintain the trustworthy connectivity and the accessibility of distributed IoT, it is important to establish secure links for end-to-end communication with a strong pervasive authentication mechanism. However, due to the resource constraints and heterogeneous characteristics of the devices, traditional authentication and key management schemes are not effective for such applications. This paper proposes a pervasive lightweight authentication and keying mechanism for WSNs in distributed IoT applications, in which the sensor nodes can establish secured links with peer sensor nodes and end-users. The established authentication scheme PAuthKey is based on implicit certificates and it provides application level end-to-end security. A comprehensive description for the scenario based behavior of the protocol is presented. With the performance evaluation and the security analysis, it is justified that the proposed scheme is viable to deploy in the resource constrained WSNs.

1. Introduction

Wireless sensor network is a key technological building block of Internet of Things, which is considered the future evolution of the Internet. During the past decade, WSN and its security are not only well investigated amongst the industry and academia [1] but also promoted with standardized security solutions [2, 3]. Although the concept and applications of IoT are not novel any longer, IoT security is still in its infancy. However, substantial amount of research work has been done to identify the challenges and possible protection mechanisms for securing IoT, as shown throughout [4–7]. Nevertheless, IoT security protocols are still neither standardized nor commercialized properly due to its novelty and immaturity. Since WSN is an indispensable part of the IoT, it needs to adapt IP technologies to create a seamless and global connectivity with the Internet [6]. The Internet engineering task force (IETF) has contributed significantly to gaining that pervasive connectivity of small objects to IPv6 based Internet. IPv6 over low-power wireless personal area network (6LoWPAN) enables complete integration of WSNs into the Internet [8, 9]. Constrained application protocol (CoAP) and routing protocol for low-power and lossy networks (RPL) are, respectively, proposed for application layer and network layer routing in constrained IoT networks [10, 11]. Physical and MAC layers of low-power networks are defined by IEEE 802.15.4 protocol [2].

In the context of IoT application domains, WSN architectures exist as centralized and distributed approaches [5]. In the centralized approach, a central entity (or a cloud service)

is responsible for acquiring raw data from the sensors, processing received data into required information and format, and providing information for other required entities (e.g., groups of companies and individual customers). In such centralized networks, there is little or no support to access the data sensing network devices directly. In contrast, the distributed networks allow the end-users and other network entities to obtain raw data straightaway from the sensor nodes. Unlike in the centralized approach, in distributed architecture the edge network devices comprise high level intelligence and processing power. Although provisioning of services is located at the edge of the network, different application platforms and end-users can collaborate dynamically with each other. As a result of the decentralized and distributed nature of the network, it is essential to consider the secure management of identity and authentication of connecting devices. In IoT applications, multiple entities (e.g., sensing nodes, service providers, and information processing systems) have to authenticate each other to establish a trusted connection. Not only should the authentication protocols be resistive and robust to malicious attacks, but also they should be lightweight to be deployed in less performing edge devices (i.e., sensors and actuators).

Rather used for generic WSN applications, IoT combined WSN use-cases are widely deployed in smart-home, smart-city, healthcare, and industry monitoring applications [5, 7, 12]. In a hospital environment, there can be different sensors installed in monitoring health conditions of patients (e.g., blood pressure, heart beat, and oxygen concentration). Doctors, who are outside the hospital, might be interested in examining health records of particular patients. Similarly, some medical machinery that maintains the environmental conditions of the ward needs to get the same records. In this scenario, as illustrated in Figure 1, doctors have to access the sensor node as an end-user and the machinery has to collaborate as a sensor node from the same or a distinctive WSN. However, in both cases, the two communication parties need to prove their authenticity to each other before establishing a secure communication link.

In factory automation and power plant monitoring applications, WSNs are deployed inside the factory premises to obtain raw data on machinery vibration, temperature, flow-rate, and light intensity [12]. Their sensed data are used to identify machine abnormalities and to create safety alarms. There can be instances where the users inside and outside the power plant want to acquire raw data directly from the sensor nodes. The end-users and the sensor nodes have to authenticate each other before transferring raw data.

Based on the explained scenarios and the state-of-the-art before, the main contributions of this paper are summarized as follows.

(i) We propose and design a pervasive authentication protocol and a key establishment scheme for the resource constrained WSNs in distributed IoT application, called PAuthKey.

(ii) We implement the PAuthKey protocol and demonstrate its performance measurements on the high resource constrained sensor nodes.

(iii) We conduct a security analysis on PAuthKey, along with performance and security comparisons between it and DTLS scheme, which is currently considered the most appropriate authentication scheme for constrained IoT networks. Moreover, we show the performance comparison results of two phases of PAuthKey with ECDSA and ECDH schemes.

The rest of the paper is organized as follows. Section 2 provides a brief overview about the related work. Section 3 comprehensively describes the system architecture, where the authentication protocol is developed, and the notations used. Section 4 presents the proposed authentication and key management protocol known as PAuthKey. Section 5 gives a detailed explanation about the implementation, performance analysis, security analysis, and scalability of the PAuthKey protocol. Finally, Section 6 concludes the paper.

2. Related Work

In centralized WSN, data from the sensor nodes are transmitted to a single central location, which processes information and combines and provides information acquisition for end-users (i.e., customers) [7]. Due to the high data availability and massive network size, processing of data on a single location might be inefficient, congested, and undertaking a high risk at single entity failure. In the distributed networks, the sensor nodes can retrieve, process, and provide data for other entities and end-users. Figure 1 provides an overview of the distributed IoT approach, which allows the communication among the edge devices, end-users, and IoT server cloud.

Distributed architecture supports the IoT network applications by providing services at local level and collaborating with all the network devices and users to achieve common goals. Due the network heterogeneity and device mobility, there can be many security threats and issues encountered with distributed IoT. In [7] Roman et al. have identified security challenges in distributed IoT. According to their study, network entity identity, authentication, access control, and secure communication channel establishment are major security concerns in distributed IoT. The proposed mechanisms should be robust to node mobility and network scalability due to the dynamic behavior of nodes. Additionally, the network needs to scale up after installation.

Exploitation of a master key for entity authentication for pervasive computing environments would be also a feasible approach to IoT enabled WSNs [13]. According to [14], the authentication mechanisms for WSN applications can be summarized as password based, remote user authentication using one-way hash functions and ticket based authentication. However, most of the work has the sole purpose of enabling end-user authentication in generic WSN architecture and it does not provide the extensibility for the key establishment. In [15, 16], the authors have proposed broadcast authentication schemes for WSNs. Reference [14] presents an effective authentication mechanism for ubiquitous collaboration in heterogeneous computing environment. This is a ticket based user authentication scheme, which is not applicable to the high resource constrained devices due to large

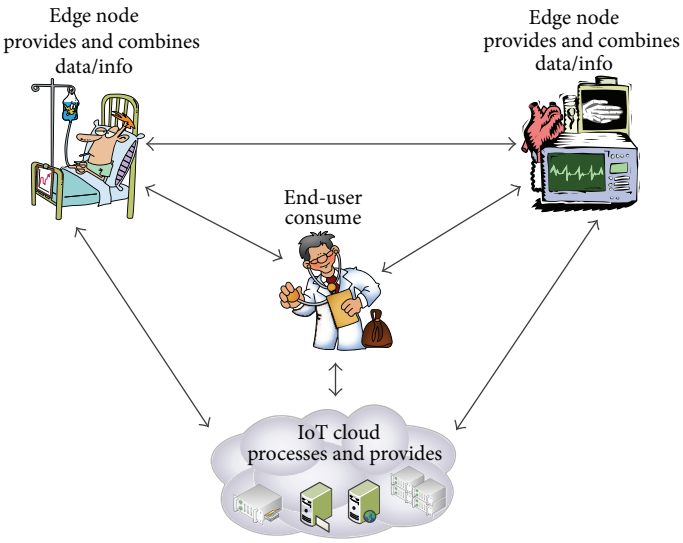

Figure 1: Overview of the distributed IoT approach in a hospital environment.

memory consumption. Nevertheless, these works have less or almost zero contribution to securing IoT combined WSNs. The reason is that they have less addressed network scalability and device mobility issues.

DTLS is an adaptation of TLS protocol and it provides an equal communication security as TLS for datagram protocols [17]. According to [10], the secured version of CoAP (known as CoAPs) is defined with DTLS due to the unreliable communication links in CoAP based IoT networks. In [18], the authors have introduced the first fully implemented two-way authentication scheme for the IoT based on DTLS protocol. However, due to the existence of eight message transfers to complete DTLS handshake, it induces a significant overhead to the network traffic. The main drawback is the utilization of X.509 certificates and RSA public keys with DTLS handshake, which are too heavy for the low performing and high resource restricted sensor nodes.

Due to the high resource demand, public key cryptographic (PKC) algorithms, such as RSA, are not recommended for WSN applications. However, elliptic curve cryptography (ECC) (i.e., a lightweight PKC alternative) based security solutions are not anymore new to WSNs. The utilization of implicit certificates for generating pairwise ephemeral keys is yet an improving realm. There are several implicit certificate generation schemes for WSNs presented in [19, 20]. Elliptic curve Qu-Vanstone (ECQV) is one of such schemes embedded in ZigBee Smart Energy applications [21]. In [22], a fully implemented end-to-end authentication scheme has been introduced to the high constrained embedded devices. TinyECC is a stable ECC implementation for constrained network entities, where in [23] the authors provide implementation details and measurement results for elliptic curve digital signature algorithm (ECDSA) and Diffie-Hellman key establishment (ECDH). Several ECC based security schemes have been proposed for WSNs as published in [15, 19, 24–26].

3. System Model and Notations

In this section, the authors provide details about the system architecture, where the protocol is modeled, and information about the used notations.

3.1. System Model. Figure 2 illustrates the assumed network architecture for the proposed authentication scheme, where end-users can collaborate with different edge devices in order to obtain particular information or service. The edge networks may include heterogeneous devices and the end-users can be humans or virtual entities (e.g., web applications).

According to the distributed IoT architecture, end-users and edge devices (i.e., sensor nodes) should possess the capability of securely accessing an edge device in a WSN. Therefore, based on Figure 2, mutual authentication is considered for four types of communication link establishments, particularly the following.

(1) Two sensor nodes are located in the same cluster (`Link A`).

(2) Two sensor nodes are located in distinctive clusters in the same WSN (`Link B`).

(3) Two sensor nodes are located in distinctive clusters and in distinctive WSNs (`Link C`).

(4) An end-user is linked to a sensor node (`Link D`).

Before starting the actual authentication protocol between two network entities, it is necessary to undergo a registration process by every communication party in order to retrieve cryptographic credentials. Later, the obtained security credentials are to be exploited for mutual authentication. For the given four types of communication link possibilities (1)–(4), every edge device and end-user have to acquire security credentials (e.g., cryptographic suites and implicit certificates) from a trusted third party such as a

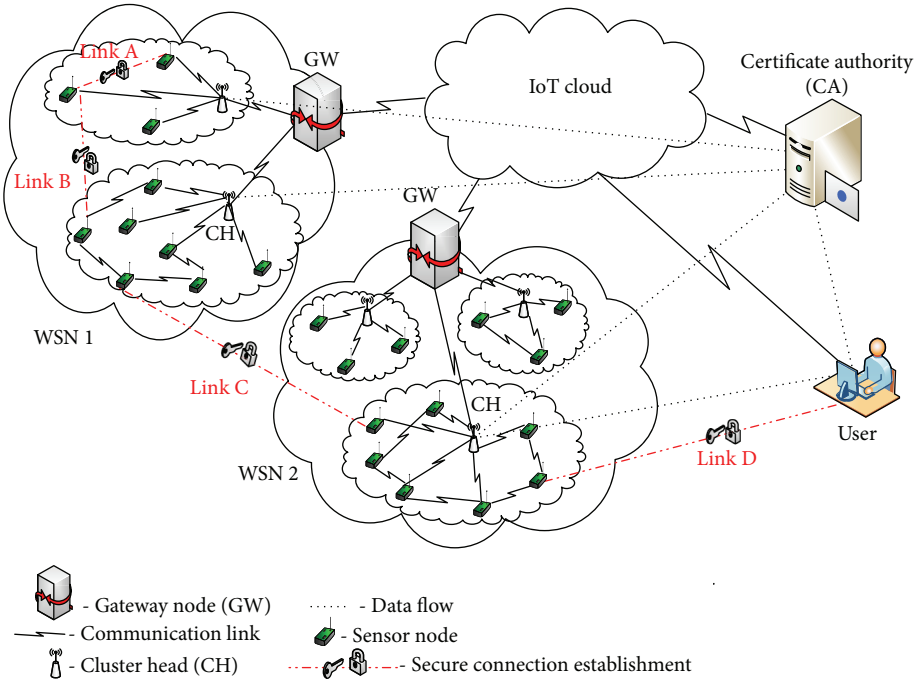

Figure 2: Assumed network architecture.

certificate authority (CA). It is assumed that the CA is a highly resource-rich server and is already known by the edge nodes during the registration phase.

In this architecture, two types of network entities such as resource rich entities (i.e., end-users and cluster heads (CH)) and highly resource constrained network entities (i.e., sensor nodes) are considered. Here, a cluster tree topology of WSNs is assumed, where a CH is the controlling device for the sensor nodes in a particular cluster. Therefore, it is considered that CH is performing as the CA (i.e., to issue implicit certificates) for the same group of sensor nodes in the cluster. However, further details about the authentication between a CH and an end-user (i.e., between two resource-rich entities) are not provided in this paper. The major concerns are the authentication between two constrained nodes or one constrained node and a resource-rich entity. Hence, it is assumed that all the CHs and end-users advocate DTLS for secure end-to-end communication after acquiring X.509 certificates from a common CA. As aforementioned in Section 2, X.509 certificates are only handled by end-users and CHs, due to their complexity and overhead on tiny sensor nodes. As illustrated in Figure 2, resource-rich network entities (i.e., end-users and CHs) first communicate with the common CA along the already established communication links heading through an IoT cloud. If an end-user needs to communicate with a sensor node with a particular cluster, first it needs to establish a secure DTLS connection with the corresponding CH and obtain implicit certificates from the CH. The end-user can obtain the implicit certificate from the CH through that secure link. Then, the end-user can use the obtained implicit certificate to communicate with the sensor node.

Having a valid implicit certificate allows the two entities for mutual authentication irrespective of their local network. Existing nodes can change their locations dynamically after requesting a new certificate. No matter what the size of the network is, adding new nodes can easily extend the data acquisition and service providing networks. It is assumed that CH can recognize the valid identities and communicate with the network entities, which are requesting security credentials [27]. The reason is that the CH has to verify the certificate requestor's identity at the beginning of the handshake and it performs the verification mainly based on the identity of the requestor node. The IPv6 over low-power wireless personal area network (6LoWPAN) identities are considered for the identification. In this paper, an end-to-end authentication is proposed for the application layer, while relying on the security schemes provided from the physical and MAC layers in IEEE802.15.4 standard [2]. Subsequently, the edge devices and end-users can mutually authenticate and establish secure communication channels, due to the distributive nature of the entire architecture.

3.2. Notations. The notations used in this paper are defined in Table 1. Elliptic curve (EC) parameters are denoted by q, a, b, G, n. The variable q is a prime, which indicates finite field F_q. The variables a and b are coefficients of EC $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \neq 0$. G is the base point generator with order of n, which is also a prime [28].

4. PAuthKey Work Flow

The PAuthKey protocol mainly consists of two phases: *registration phase* and *authentication phase*. During the registration phase, the sensor nodes in a particular cluster should obtain certificates from the CH and derive their own public-private key pairs. The authentication phase is varying upon

Table 1: Notations used in cryptographic algorithms.

Notations	Description
K	Network-wide symmetric key for initial message authentication
r_U	Secret random integer value generated by U
R_U	EC point for certificate request sent by node U
Cert_U	Implicit certificate of Uth node
e	Integer used to keep hash value of Cert_U
s	Integer used to compute private key of the requestor node
d_U	Node U's private key
Q_U	Node U's public key
N_U	Random cryptographic nonce generated by node U
K_{UV}	Link key between nodes U and V

the type of the communication link between the end-parties (i.e., four communication link possibilities as explained in Section 3). Accordingly, the authentication phase is described for three scenarios with reference to the system model in Figure 2: scenario 1 for link A, scenario 2 for links B and C, and scenario 3 for link D. The upcoming subsections characterize the phases individually.

4.1. Registration Phase (Initial Certificate Acquisition). Initially, the sensor nodes should obtain security credentials from their respective cluster head (CH) as a prerequisite for the actual authentication protocol. All the sensors in a particular cluster consider their certificate authority (CA) as the CH. Upon the certificate request from sensor node U, the CA generates the certificate. The message flow of the certificate acquisition is illustrated in Figure 3. The grey boxes show the change of variables and the white boxes indicate the performed functionality by the entity.

The handshake starts with the `Requestor Hello` message, 6LoWPAN node identity (U), and cipher suites that are supported by the requestor. The cipher suites are common for all the edge devices and would include the available cipher options on the requestor, for example, *CERT_ECC_160_WITH_AES_128_SHA1* requests for certificates in 160-bit EC curves, 128-bit AES for bulk encryption, and SHA1 for hashing. It is assumed that the cipher suites are installed on the sensor nodes during the offline mode before the deployment. CA uses 6LoWPAN node identity to figure out whether it is a valid request from a node that belongs to its cluster. If the requestor identity verification is successful, CA agrees to one cipher suite from the received options and sends `CA Hello` message with its public key Q_{CA} as an unprotected message to approve the initiation of the handshake.

Upon receiving `CA Hello` message, the requestor generates a certificate request EC point R_U. While creating a certificate request, first, the node generates a random number $r_U \in [1, \ldots, n-1]$ and computes $R_U = r_U \times G$. The predefined EC domain parameters are used according to the negotiated cipher suite. Second, the node produces a random cryptographic nonce N_U, calculates message authentication code (MAC) value (i.e., $\text{MAC}[R_U, U, N_U]$), and sends those two along with the `Certificate Request` message. The CA generates the MAC value using the common message authentication key K, which is mentioned in the cipher suite. The random cryptographic nonce is used to ensure the freshness of the message.

After receiving the certificate request, CA verifies the MAC value and nonce N_U in order to identify the integrity and the freshness of it. If the verification is successful, CA computes the certificate Cert_U and private key reconstruction value s for the requestor node U. During this process, CA first generates a random number $r_{\text{CA}} \in [1, \ldots, n-1]$ and computes the certificate $\text{Cert}_U = R_U + r_{\text{CA}} \times G$. Then, CA calculates s using Cert_U, r_{CA}, and its own private key d_{CA}; $e = H(\text{Cert}_U)$ and $s = er_{\text{CA}} + d_{\text{CA}} (\text{mod } n)$. The value e should be computed using the one-way cryptographic hashing function, which is mentioned in the cipher suite (e.g., SHA1). Later, the CA sends the `Certificate` message that includes the certificate Cert_U, s, a random nonce N_{CA}, and the MAC on $[\text{Cert}_U, s, N_{\text{CA}}]$.

While receiving this message, the requestor node U first verifies the MAC and N_{CA}. If they are correct, U calculates $e = H(\text{Cert}_U)$ using the same hash function. Then, the node U can compute its own private key $d_U = er_U + s (\text{mod } n)$ and public key $Q_U = d_U \times G$.

Node U's `Finished` message contains an encrypted message digest of previous handshake messages using the requestor public key Q_U. According to the EC arithmetic operations which are performed for calculating keys [10], CA is also capable of computing U's public key Q_U; $Q_U = e\text{Cert}_U + Q_{\text{CA}}$. The following derivation proves that both equations give exactly the same Q_U as computed by node U:

$$\begin{aligned} Q_U = d_U G &= \left(er_U + s \,(\text{mod } n)\right) G \\ &= \left(er_U + er_{\text{CA}} + d_{CA} \,(\text{mod } n)\right) G \\ &= e\left(r_U + r_{\text{CA}} \,(\text{mod } n)\right) G + d_{\text{CA}} G \\ &= e\left(r_U G + r_{\text{CA}} G\right) + Q_{\text{CA}} \\ &= e\left(R_U + r_{\text{CA}} G\right) + Q_{\text{CA}} \\ &= e\text{Cert}_U + Q_{\text{CA}}. \end{aligned} \tag{1}$$

CA uses public key Q_U for encrypting previous messages and answers with the `Finished` message to complete the handshake of the preauthentication phase.

At the end of the registration phase, the sensor nodes possess the security credentials to start secure communication with the internal and the external network entities (i.e., end-users and sensor nodes).

4.2. Authentication Phase. The authentication phase is described for three scenarios. Scenario 1 (`Link A`) is the authentication between two sensor nodes in the same cluster. Parts of scenario 2 (`Link B and C`) and scenario 3 (`Link D`) also include the principal handshake of scenario 1. Scenario 2 is the authentication between two sensor nodes in distinctive clusters in the same or different WSN. Scenario 3 is the authentication between a sensor node and an end-user. These

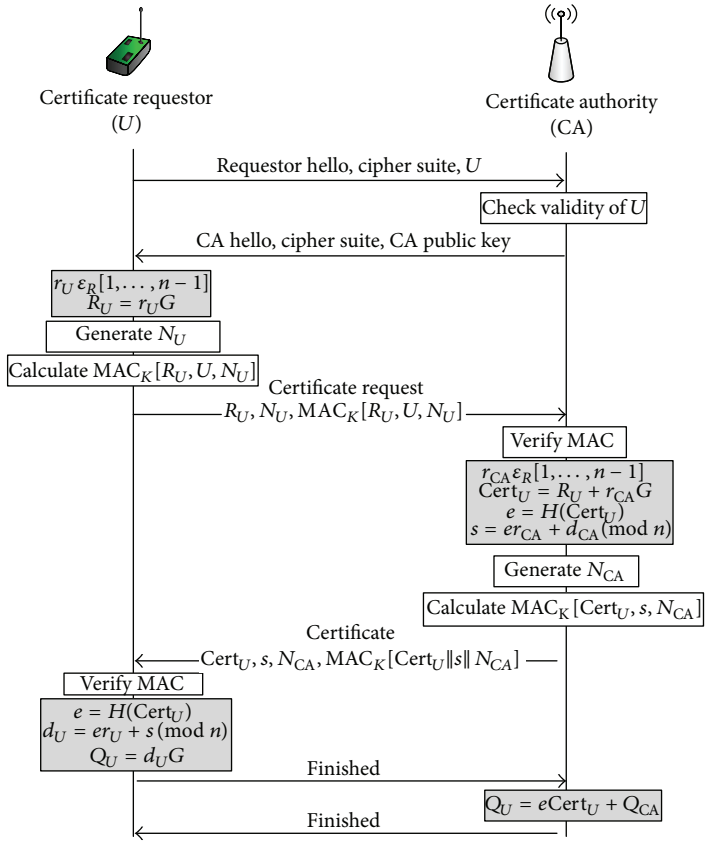

Figure 3: Message flow for the registration phase.

scenarios resemble the communication link possibilities, which are shown in Figure 2.

Scenario 1: Authentication Process between Two Sensor Nodes in the Same Cluster. The first scenario is the exploitation of certificates and public-private keys for node authentication between two sensor nodes in the same cluster. Since the sensor nodes in a particular cluster obtain security credentials from a common CA, they can easily carry out the mutual authentication as depicted in Figure 4. The grey boxes are value ranges and the white boxes are performed operations. The client node U is aware of the 6LoWPAN identity of the server node V, which U needs to acquire the data or the service. As the initial step, the client sends the `Client Hello` message accompanied with its identity U to the server node. The server replies with the `Server Hello` message along with the cipher suites it supports and CH's identity, which is the CA for both nodes. If the client does not have the security credentials from the given cipher suite, it has to retrieve them from the CA. Otherwise, the client can continue the handshake by sending its certificate Cert_U. Similar to the registration phase handshake, random cryptographic nonce N_U and MAC values are used in order to preserve the freshness and integrity of the message.

Upon receiving the client's certificate, the server first verifies the MAC value and then computes the client's public key Q_U using its certificate; $e = H(\text{Cert}_U)$ and $Q_U = e\text{Cert}_U + Q_{\text{CA}}$. This is proven according to (1). Then, the server V generates a random nonce N_V and sends it along with Cert_V and $\text{MAC}_K[\text{Cert}_V, N_V, V]$. In the meantime, the server V computes the pairwise key K_{UV} from its private key d_V and U's public key Q_U, where $K_{UV} = d_V Q_U$. Similar to the server V, upon receiving the message, the client U verifies the MAC and if the verification is successful it computes Q_V and $K_{UV} = d_U Q_V$. Therefore, at the end of two-way message transferring, both parties can derive a common pairwise key for actual secure communication.

Finally, the exchange of the `Finished` messages concludes the handshake. This Finished message is composed of previous handshake messages, which are encrypted by the common key K_{UV}. At the end of six message transfers, the two edge nodes can authenticate each other and establish a common secret key and a secure communication link that can be used for securing further data acquisitions between the client and the server.

Scenario 2: Authentication Process between Two Sensor Nodes in Distinctive Clusters. Here, the node authentication process is demonstrated between two sensor nodes located in distinctive clusters, which might be in the same or different WSNs. In such cases, the nodes cannot use their certificates for mutual authentication since they are generated from two CAs. The messages flow of the authentication protocol is illustrated in Figure 5.

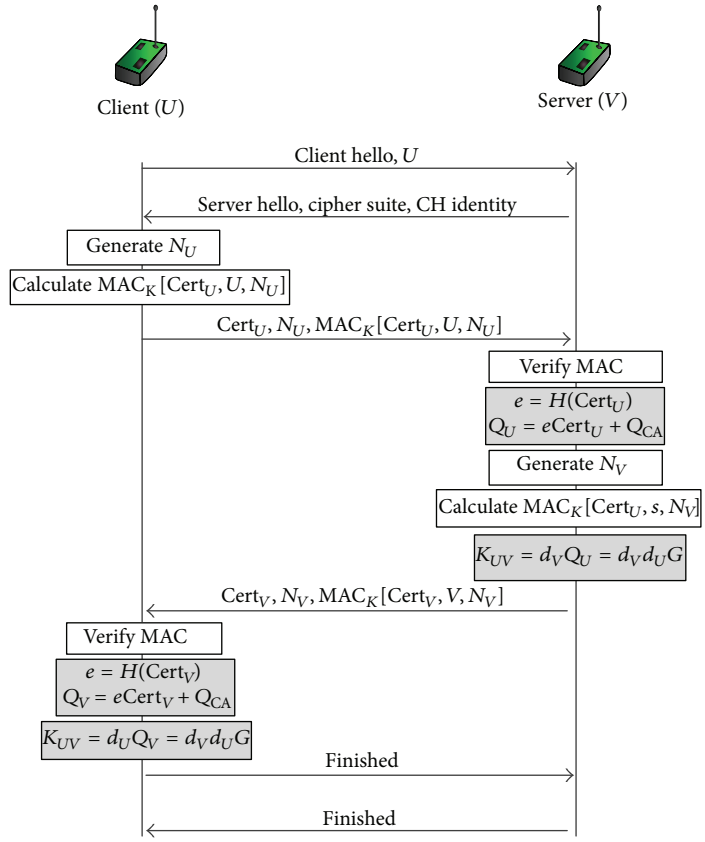

Figure 4: Message flow for scenario 1—authentication process between two sensor nodes in the same cluster.

Similar to scenario 1, the preliminary `Hello` message exchange initiates the handshake. Upon receiving the `Server Hello` message, the client checks the identity of the CH corresponding to the server and verifies the necessity of requesting a new certificate from the given CH. The client U is unable to communicate with CH_V directly and it obtains the security credentials through its own cluster head CH_U. The client forwards the cipher suite and *CH identity* to its cluster head CH_U. CH_U communicates with CH_V on behalf of the node U and requests the security credentials. As explained in the system model, it is assumed that two cluster heads initiate a secure communication link using RSA keys and DTLS handshake [18]. Though CH_V does not communicate with the node U directly, it grants the certificate by being the CA for the node U. The new certificate request and the certificate are exchanged through that established secure channel. Additionally, CH_V sends its public key Q^*_{CA} to CH_U, which is used by U during further computations. Then, CH_U computes the new public-private keys Q^*_U and d^*_U of node U and sends the certificate Cert^*_U and the keys (i.e., Q^*_U, d^*_U, and Q^*_{CA}) to the node U. The security credentials are encrypted by U's primal public key Q_U. After having the required security credentials from the new CA (i.e., cluster head of node V), the client U can continue the authentication protocol as described in scenario 1 handshake.

Scenario 3: Authentication Process between End-User and Sensor Node. Figure 6 demonstrates the flow of the message transactions of the authentication process between an end-user and a sensor node. The difference between scenario 3 and scenario 2 is that here the user directly retrieves security credentials from the given cluster head. However, in scenario 2, the two cluster heads have to communicate first for the acquisition of the implicit certificate for the client node. Similar to the previous case (i.e., scenario 2), it is assumed that the secure link between the user and the CH is established with RSA keys and DTLS handshake [18], and the security credentials are transmitted over that link. Once the end-user obtains the certificate and computes its public-private keys, the rest of the handshake would occur in a similar manner as scenarios 1 and 2.

As explained in the above three scenarios, the end-users and the sensor nodes can establish secure communication links after authenticating each other using implicit certificates. Furthermore, the two-party authentication mechanism enables the nodes to generate a pairwise common secret key. Therefore, this would advocate accessing the data and services in WSNs accommodated in distributed IoT architecture.

5. Analytical Justification of PAuthKey

In this section, a comprehensive analysis of the proposed PAuthKey protocol is presented. The performance analysis is given in terms of memory, energy consumption, and execution time, along with the support for network scalability.

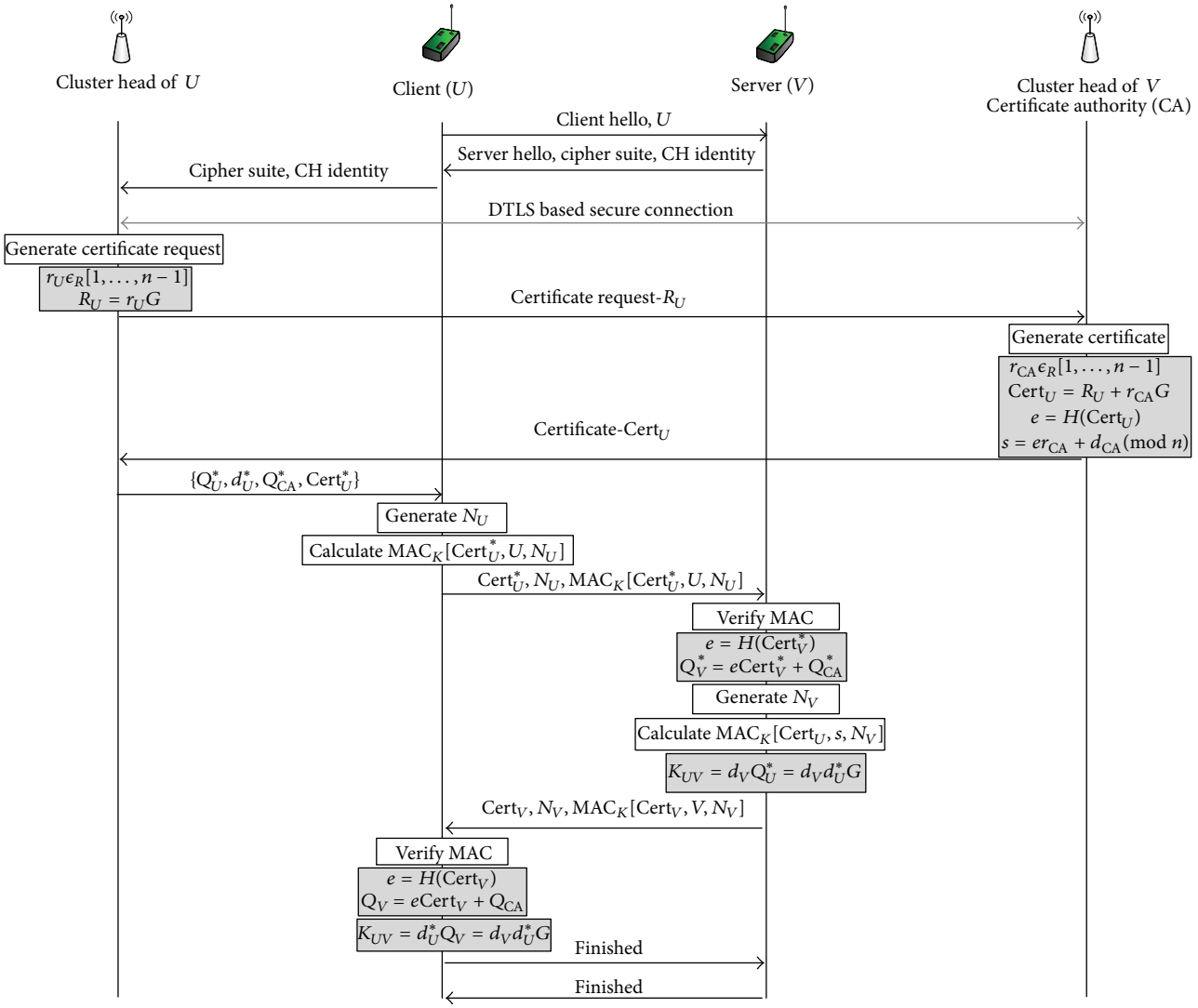

Figure 5: Message flow for scenario 2—authentication process between two sensor nodes in distinctive clusters.

Next, the security of the protocol and the comparisons with the related work are discussed.

5.1. Performance Analysis. Physical and MAC layer security protocols do not provide end-to-end communication security. DTLS is the widely used application level security protocol for authentication in IoT networks. Variants of DTLS handshakes are based on ECC and used with RSA and X.509 certificates [22]. Although the exploitations of RSA and X.509 certificates with DTLS provide interoperability, they are hardly utilized by the high resource constrained devices (e.g., sensors). The major drawbacks are as follows.

(1) RSA has a key size of 2048 bits.

(2) Standard X.509 certificates are in the order of 1 kB in size.

(3) The utilization of RSA and X.509 on constrained sensors consumes resources and induces computation overhead.

The PAuthKey solution is implemented on a simple network with TelosB sensor nodes [29] that have IEEE 802.15.4 compliant CC2420 RF transceivers. The hardware includes 8 MHz, 16-bit MCU with 10 Kbyte RAM and 48 Kbyte ROM. CC2420 RF transceiver has a maximum data rate of 250 kbps and frequency band of 2400 MHz [29]. PAuthKey is developed in NesC on TinyOS 2.1.2 [30]. ECC (i.e., for EC arithmetic operations) and natural number (NN) (i.e., for large natural number operations) interfaces are utilized from TinyECC configurable library [23]. *secp160r1* EC domain parameters are used as defined in [28]. The authors of this paper utilized EC optimization techniques provided in TinyECC such as *Barrett reduction* to speed up modulo operations, *Hybrid Multiplication* and *Squaring* for integer multiplication, *Projective Coordinate Systems* for point addition, and *Sliding Window* for scalar multiplication. SHA-1 is used as the one-way cryptographic function H. ECC operations are extremely costly compared to other cryptographic operations (i.e., SHA-1 and MAC) [23]. Therefore, we have considered the given EC operation optimization techniques.

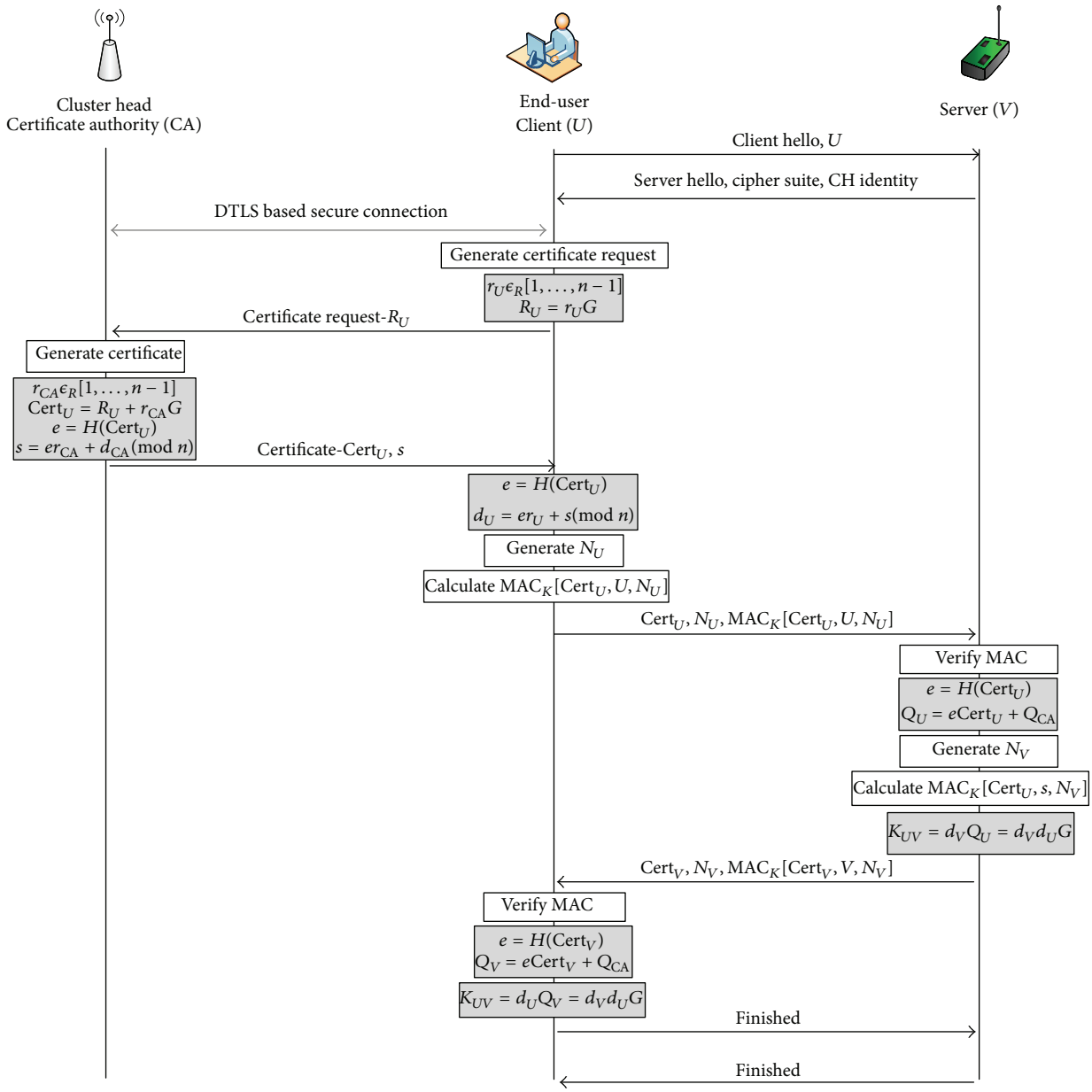

Figure 6: Message flow for scenario 3—authentication process between end-user and sensor node.

The experimental setup comprises three TelosB nodes, one as the CA and the rest as the cluster nodes. For the sake of simplicity and comparison, CA functionalities are also implemented on the sensor node itself. The measurements are taken in terms of execution time, energy, and memory (i.e., RAM and ROM) consumption. The *check_size.pl* script is used to obtain memory consumption values (e.g., for RAM and ROM) required by each operation in registration phase and authentication phase for scenario 1. The execution times are measured directly on the sensor nodes and the energy consumptions are computed using the runtime. The energy consumptions are then calculated as $V \times I \times t$ based on the voltage (V), the current (I), and the execution time (t) on TelosB sensor nodes [29]. Similar to [23], it is also considered that the static values given in [29] are $V = 3$ volts and $I = 1.8$ mA.

Table 2: Memory utilization.

	RAM (bytes)	ROM (bytes)
Registration phase		
Edge node operations	1410	11774
CA operations	2332	16576
Authentication phase		
Operations at one edge node for scenario 1	1530	11650

As given in Table 2, memory utilization values are taken for two phases with respect to the communicating nodes.

Table 3: Execution time and energy consumption.

Operation	Time (ms)	Energy (mJ)
Registration phase		
At the sensor node side		
Initialization	2709	14.6286
Cert Req generation	2764	14.9256
Cert verification	2758	14.8932
`Finished` msg computation	4	0.0216
At the CA side		
Initialization	2709	14.6286
Cert generation	5728	30.9312
`Finished` msg computation	2154	11.6316
Authentication phase (scenario 1)		
At the client or the server side		
Initialization	2672	14.4288
Key computation	5768	31.1472
`Finished` msg computation	4	0.0216

For the registration phase, the total memory consumption is measured for edge node operations (i.e., certificate requestor) and CA operations. Edge node operations include the generations of `Requestor Hello`, `Certificate Request`, and `Finished` messages, certificate verification, and public-private key computation at the sensor node side. Similarly, CA operations include the generation of `CA Hello`, `Certificate`, and `Finished` messages and `Certificate Request` verification.

For the authentication phase, scenario 1 was only considered, since it is almost similar to the major overhead created at the sensor node side for the other scenarios. According to the message flow of the authentication phase (i.e., scenario 1 in Figure 4), the collective operations performed at the client and the server sides are identical. Therefore, the operations at one edge node include the generation of `Hello` and `Finished` messages, MAC computation and verification, public key calculation, and the derivation of the common secret key. As a result, the two phases of the protocol collaboratively consume 2940 bytes of RAM and 23424 bytes of ROM. However, the overall implementation of two phases is still below the 10 kB RAM and 48 kB ROM provided by the high resource constrained TelosB sensor nodes. Although the memory consumption is higher for CA operations, in the real-time deployment it would be tolerable for a resource-rich device.

Since the transmission time depends on the size of the network and the distance between the nodes, only the execution time for the particular operations performed at the edge nodes and CA is measured for the registration and authentication phases for scenario 1. The measured execution time values and the calculated energy consumption values are depicted in Table 3.

During the registration phase, the approximate collective time utilization at certificate requestor's (i.e., the sensor node) side is 8235 ms. This value includes the execution times for initialization (2709 ms), certificate request generation (2764 ms), certificate verification (i.e., private-public key derivation) (2758 ms), and Finished message computation (4 ms). At CA's side, the execution time values are taken for initialization (2709 ms), certificate generation (5728 ms), and `Finished` message computation (2154 ms). Altogether, CA spends 10591 ms for its operations under the registration phase. At CA's side, the `Finished` message computation has a higher execution time due to the derivation of the public key of the sensor node (i.e., EC operation $Q_U = e\text{Cert}_U + Q_{CA}$).

During the authentication phase for scenario 1, each edge node (i.e., the client or the server) takes approximately 8444 ms for initialization, key computation, and `Finished` message computation. For the same operations, we have calculated the energy consumptions at TelosB nodes using $V \times I \times t$. According to the computed values, a TelosB sensor node consumes nearly 44.47 mJ and 45.6 mJ for registration and authentication phases (for scenario 1), respectively. However, these timing, energy, and memory values can be improved by using further optimized basic ECC arithmetic operations. All in all, experimental results show that the proposed authentication mechanism can be easily deployed in low-power less performing devices.

In the proposed two-phase authentication protocol, implicit certificates, which are 160-bit EC points instead of X.509 certificates, were used. Therefore, the size of the certificate is only 44 bytes. Using optimally designed EC curves we can reduce the certificate size and using compression techniques we can further decrease the overall message size. Retransmission clocks can be used at both communicating parties for identifying timeouts and retransmitting when there is a message loss. Furthermore, the authentication protocol supports scaling up the network, since the newly added nodes can authenticate themselves after undergoing the registration phase. As the certificates are not based on the physical locations of the edge devices, they do not have to be alternated according to nodes' mobility.

5.2. Scalability. The proposed authentication protocol supports the scalability of the network (i.e., expanding the network with new node addition) and the location changes of the sensor nodes within the same cluster. When a new node is added to the network, a valid 6LoWPAN node identity, K message authentication key, and cipher suites should be stored while the node is at offline mode. Figure 7 illustrates how PAuthKey protocol supports a new node addition to the network, within a particular cluster. It is illustrated as a three-stage process. In *Stage 1*, at the bootstrapping phase, the newly added node (marked as red rectangular shaped) can send the certificate request and obtain a certificate from the CA for computing its own keys. Hence, the size of the network is not necessary to be predefined during the initial design phase and deployment phase. At a new node request, the CA only needs to verify the validity of the sensor node identities to issue the certificate. In *Stage 2*, the new node receives its certificate. Therefore, *Stages 1* and *2* resemble the registration phase of PAuthKey protocol. Finally in *Stage 3*, the node can undergo the authentication phase and the key establishment using the received certificate.

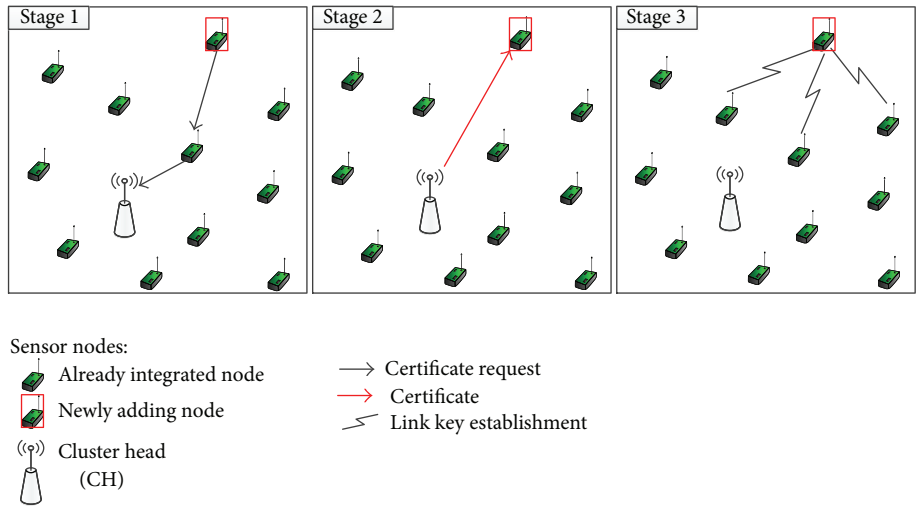

FIGURE 7: Behavior of the protocol when a new sensor node enters the cluster.

Similarly, the sensor nodes do not need prior knowledge of their neighbors. Whenever a new node is added to the network or changes the neighboring set, it can establish the ephemeral pairwise link keys with the corresponding neighbors using the certificate. The certificates always provide an implicit assurance for the sensor nodes that they are legitimate nodes. Even though the sensor nodes are frequently changing their locations (i.e., also the neighboring set), they can authenticate themselves and derive the pairwise keys securely without previous awareness of the new neighboring nodes or end-users. As shown in [31], if the authentication is performed based on pairwise keys between neighbors which are preinstalled, then there should be a large number of stored keys per node, which may not be desirable for large scale networks. However, in PAuthKey protocol, such a large scale key preinstallation is not needed at all since the ephemeral link keys have to be established before starting communication.

5.3. Security Analysis. The proposed implicit certificate based authentication protocol is developed using one of the lightest PKC schemes, ECC. Though it is comparatively more expensive than symmetric key algorithms, it is inherently secured due to the PKC characteristics. However, as shown in Sections 5.1 and 5.2, the proposed scheme is feasible to deploy in real-time WSNs. While using EC scalar-point multiplication, the scheme is provably secured under the random oracle model that the discrete logarithm problem over the subgroup is intractable. The advantage of using ECC is that it provides an equal security as RSA, however, with less overhead (e.g., 160-bit ECC equals RSA with 1024 key size). At the end of the authentication scheme there is a key establishment part which extends the security strength of the standard ECDH key agreement by using mutually authenticated keying materials (i.e., Cert_U and s). Since the authentication is performed using the implicit certificates of edge devices or users and the link keys are derived using two preevaluated values (i.e., d_U and Q_V), the legitimacy and trust between two parties are implicitly assured.

It is very common that denial of service (DoS) attacks can be launched against distributed IoT. Moreover, during the registration phase, the first `Hello` message contains the certificate requestor's identity, which is analyzed by CA. If the unauthorized requestors are trying to access, the CA can identify them at the beginning of identity verification and protect itself from DoS attacks. Similarly, during the authentication phase, the cryptographic credentials are exchanged only after the successful `Hello` message exchange, which provides protection from DoS attacks. In order to overcome illegal message alternations by malicious users and DoS attacks, the subsequent messages contain MAC with the common authentication key K for preserving data integrity. The availability of the proposed protocol is ensured by giving permission to two legitimate nodes, which possess the certificates granted from the CA, to authenticate each other and establish a secure pairwise key for their mutual communication. The freshness of the messages is guaranteed by appending true nonce. In the registration phase, the originator of the certificate (i.e., CA) cannot deny being sent the message (i.e., nonrepudiation property), since the CA uses its key pair to generate the certificate and private key reconstruction value (s). Likewise, during the authentication phase, the sender of the messages cannot deny that the messages are sent by itself since the receiver always uses the certificate of the sender to derive its (i.e., the sender's) public key.

In the security analysis, we are considering three attacks including node compromising attacks, masquerade attacks, and impersonate attacks. In node compromising attacks, an adversary can physically capture a node and obtain its keys. Similarly, in the authentication phase, an attacker can impersonate a legitimate sensor node using its certificate or try to masquerade the key establishment between two legitimate nodes.

5.3.1. Node Compromise Attacks. PAuthKey is resilient to node compromise attacks. If a sensor node U is captured, the adversary can reveal Cert_U, Q_U, d_U, and Q_{CA}. However,

with CA's public key, an adversary cannot create a new valid certificate and private key reconstruction data, since they are derived using d_{CA}, which is only known to CA. Though a node pretends to be a forgery CA and issues certificates with Q_{CA}, eventually the fake certificates and public keys are disclosed during the key establishment in the authentication phase (i.e., calculating $K_{UV} = d_U Q_V$). We assume that the CH can identify compromised nodes using beacon message technique, as explained in [20, 32]. Then, CA will broadcast the compromised node ID to the noncompromised nodes within its cluster and the other CHs and end-users, those who have contacted the CA for acquiring certificates to communicate with that compromised node. Upon receiving the CA's message (i.e., compromised node ID), the other sensor nodes, CHs, and end-users will discard or order to demolish the certificate of the corresponding node and the preestablished pairwise keys. Then, the compromised node cannot appear by itself as a legitimate node in the future because its certificate and user ID are already abandoned by the legitimate users.

5.3.2. Impersonation and Masquerade Attacks. During the key establishment in the authentication phase, nodes are authenticated in order to prevent impersonation attacks and masquerade attacks. Node V computes node U's public key using its Cert_U and the CA's public key Q_{CA}. The pairwise key K_{UV} calculation at both ends will be the same, only if the certificates are issued by the identical valid CA. Therefore, node V has an implicit assurance that the received certificate is genuine (i.e., issued by the CA). Likewise, when two legitimate nodes initiate a pairwise key, an attacker (without a valid certificate) cannot come in between them and masquerade the key establishment. Since the pairwise key is derived on the basis of certificates and private keys of both parties, an attacker is impossible to proceed in it by only using a valid certificate.

5.4. Comparison with Related Work. In this paper, the focus was on authenticating the extreme resource constrained devices, which are deployed in WSNs in distributed IoT applications. Therefore, the proposed authentication protocol was implemented on TelosB sensor nodes and performance measurements were obtained. However, as explained in Section 2, DTLS is considered the prominent authentication protocol for IoT applications. Though we use DTLS in the middle of PAuthKey protocol for certain scenarios (i.e., scenarios 2 and 3), the key foundation of the proposed authentication scheme is explained in scenario 1. In particular, in scenarios 2 and 3, DTLS is also utilized by resource-rich entities such as CHs and end-users for authentication.

As aforementioned in Sections 2 and 4, PAuthKey scheme is inspired by different ECC based security schemes. Among them, ECDSA and ECDH are the most relevant schemes to two phases of PAuthKey protocol. Therefore, the first assessment includes the assessment of PAuthKey scheme with the related work as depicted in Table 4. The memory and timing values of ECDSA digital signature scheme are compared with those of the registrations phase. Similarly, ECDH key establishment performance is contrasted with the key computation of the authentication phase. All the empirical results are measured on TelosB sensor nodes and with the activation ECC optimization techniques as mentioned in Section 5.1.

Table 4: Comparison of PAuthKey, ECDSA, and ECDH schemes.

	RAM (bytes)	ROM (bytes)	Time (ms)
Registration phase			
At the sensor node side	1410	11774	8231
At the CA side	2332	16576	8437
ECDSA scheme [23]	1586	12640	14789
Authentication phase (scenario 1)			
Key computation	1530	11650	5768
ECDH scheme [23]	1866	12102	6146

Table 5: Comparison of PAuthKey and DTLS scheme.

	DTLS scheme [18]	PAuthKey scheme (for scenario 1)
Memory consumption		
ROM	67 kB	22.875 kB
RAM	20 kB	2.871 kB
Time for authentication	4000 ms	8444 ms
Energy for authentication	939 mj	45.59 mj
Key size	2048 bits (RSA)	160 bits (ECC)

According to the given experimental results, the registration phase of PAuthKey scheme at the sensor node side consumes less memory than ECDSA scheme. Although the proposed scheme consumes higher memory values than ECDSA scheme, it would be tolerable for a resource-rich device. However, the execution times of registration phase at both ends (i.e., sensor node and CA) are less than the conventional ECDSA scheme. Similarly, the key computation of the authentication phase utilizes less memory and time than the ECDH scheme. In security aspects, conventional ECDH scheme is vulnerable to impersonation and masquerade attacks, since two communication parties do not have an authentication phase during the key establishment. However, the proposed key establishment is well secured at both types of attacks. Therefore, the given comparison results witness higher performing capability of PAuthKey scheme in the resource constrained sensor nodes than ECDSA and ECDH schemes.

The second assessment presents the comparison results between PAuthKey scheme and conventional DTLS scheme. Thereby, the appropriateness of the proposed protocol for the high resource restricted sensor nodes in WSNs is shown. We use the empirical values, which indicate the performance of DTLS, as given in [18] and the experimental results for PAuthKey. Table 5 shows the comparison results between DTLS scheme and PAuthKey authentication mechanism (i.e., for scenario 1).

The memory utilizations of PAuthKey are much better than the conventional DTLS scheme. This is a convincing remark, which confirms the applicability of PAuthKey scheme for the high resource constrained sensor nodes. Similarly, energy consumption for the authentication in PAuthKey scheme is notably fitting with low-power devices. According to the experimental results, the total time consumption for PAuthKey authentication is nearly double the value of DTLS authentication. However, this can be further reduced by using optimized EC arithmetic operations. Therefore, the authors of this paper believe that the proposed solution PAuthKey extends the existing pool of security solutions concerned with ECC and can optimize the key establishment in WSNs.

6. Conclusion

In this paper, the authors have introduced and analyzed an authentication and key establishment mechanism for WSNs in distributed IoT applications. The proposed PAuthKey protocol comprises two phases: *registration phase* for obtaining cryptographic credentials to the edge devices and end-users and *authentication phase* for authentication and key establishment in mutual communication. The authentication phase is described for three distinctive scenarios, based on the links between two communicating parties. Using PAuthKey protocols, the end-users can authenticate themselves to the sensor nodes directly and acquire sensed data and services. With the experimental results, it is shown that the authentication protocol is feasible to deploy in the low performing resource constrained network devices in WSNs. The protocol supports the distributed IoT applications, since the certificates are lightweight and can be handled by the high resource constrained devices, irrespective of their originality. According to the security analysis, the PAuthKey scheme is secured under certain types of attacks. Finally, a brief comparison between the conventional DTLS scheme and the proposed PAuthKey protocol is presented. This shows the appropriateness of PAuthKey scheme especially on the high resource constrained devices.

In the future, the authors intend to extend the utilization of implicit certificates for access control and multicasting in the massive scale distributed IoT network applications. It is expected to customize the content of the implicit certificates by adding other information, such as the time stamp, location identity, or 6LoWPAN identity, depending upon the application requirements. Furthermore, it is intended to extend the utilization of implicit certificates for group key management in large scale sensor networks.

Disclosure

Part of this work is published at the 10th IEEE International Conference on Embedded Software and Systems, 2013 [33], and the 14th IEEE Wireless Communication and Networking Conference, 2014 [34]. The extensions of this work include the authentication protocol between sensor nodes and the implementation and evaluation of PAuthKey protocol.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work has been supported by Tekes under Massive Scale Machine-to-Machine Service (MAMMotH) project and Academy of Finland project SEMOHealth. Pawani Porambage is also supported by HPY research foundation scholarship granted from Elissa Cooperation, Finland.

References

[1] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: a survey," in *Distributed, Grid, and Pervasive Computing*, X. Yang, Ed., p. 849, CRC Press, 2007.

[2] "IEEE Standard for Low-Rate Wireless Personal Area Networks (LRWPANs)," IEEE Std 802.15.4. 2011 (Revision of IEEE Std 802.15.4- 2006), 2011.

[3] "ZigBee Specification Version 1.0," ZigBee Alliance, 2008, http://www.zigbee.org/home.aspx.

[4] R. H. Weber, "Internet of things—new security and privacy challenges," *Computer Law and Security Review*, vol. 26, no. 1, pp. 23–30, 2010.

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[6] M. Brachmann, S. L. Keoh, O. G. Morchon, and S. S. Kumar, "End-to-end transport security in the IP-based internet of things," in *Proceedings of the 21st International Conference on Computer Communications and Networks (ICCCN '12)*, pp. 1–5, IEEE, Munich, Germany, August 2012.

[7] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[8] N. Kushalnagar, G. Montenegro, and C. Schumacher, " IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): overview, Assumptions, Problem Statement, and Goals," IETF RFC 4919, 2007, http://tools.ietf.org/html/rfc4919.

[9] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," IETF R FC 4944, September 2007, http://tools.ietf.org/html/rfc4944.

[10] Z. Shelby, K. Hartke, and C. Bormann, "Constrained Application Protocol (CoAP)," IETF draft, R FC editor, 2013, http://tools.ietf.org/pdf/draft-ietf-core-coap-18.pdf.

[11] T. Winter, P. Thubert, A. Brandt et al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, 2012, http://tools.ietf.org/html/rfc6550.

[12] K. Islam, W. Shen, and X. Wang, "Wireless sensor network reliability and security in factory automation: a survey," *IEEE Transactions on Systems, Man and Cybernetics C: Applications and Reviews*, vol. 42, no. 6, pp. 1243–1256, 2012.

[13] F. Zhu, M. W. Mutka, and L. M. Ni, "Private entity authentication for pervasive computing environments," *International Journal of Network Security*, vol. 14, no. 2, pp. 86–100, 2012.

[14] S. Shin, T. Shon, H. Yeh, and K. Kim, "An effective authentication mechanism for ubiquitous collaboration in heterogeneous

computing environment," *Peer-to-Peer Networking and Applications*, 2013.

[15] Y. Liu, J. Li, and M. Guizani, "PKC based broadcast authentication using signature amortization for WSNs," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 2106–2115, 2012.

[16] T. Kwon and J. Hong, "Secure and efficient broadcast authentication in wireless sensor networks," *IEEE Transactions on Computers*, vol. 59, no. 8, pp. 1120–1133, 2010.

[17] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," IETF RFC 4347, April 2006, http://tools.ietf.org/html/http://tools.ietf.org/html/rfc4347.

[18] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "DTLS based security and two-way authentication for the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710–2723, 2013.

[19] K. Malasri and L. Wang, "Design and implementation of a secure wireless mote-based medical sensor network," *Sensors*, vol. 9, no. 8, pp. 6273–6297, 2009.

[20] R. Lu, X. Li, X. Liang, X. Shen, and X. Lin, "GRS: the green, reliability, and security of emerging machine to machine communications," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 28–35, 2011.

[21] "SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), Version 0.97," August 2013, http://www.secg.org/.

[22] V. Gupta, M. Wurm, Y. Zhu et al., "Sizzle: a standards-based end-to-end security architecture for the embedded Internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425–445, 2005.

[23] A. Liu and P. Ning, "TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 245–256, St. Louis, Mo, USA, April 2008.

[24] X. H. Le, S. Lee, I. Butun et al., "An energy-efficient access control scheme for wireless sensor networks based on elliptic curve cryptography," *Journal of Communications and Networks*, vol. 11, no. 6, pp. 599–606, 2009.

[25] C. T. Li, M. S. Hwang, and Y. P. Chu, "An efficient sensor-to-sensor authenticated path-key establishment scheme for secure communications in wireless sensor networks," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 8, pp. 2107–2124, 2009.

[26] P. Kotzanikolaou and E. Magkos, "Hybrid key establishment for multiphase self-organized sensor networks," in *Proceedings of the 6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM '05)*, pp. 581–587, June 2005.

[27] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, "Toward trusted wireless sensor Networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 1, article 5, 2010.

[28] D. Hankerson, S. Vanstone, and A. J. Menezes, *Guide to Elliptic Curve Cryptography*, Springer, New York, NY, USA, 2004.

[29] "TelosB Datasheet," Tech. Rep., Crossbow Inc., 2013, http://www.datasheetarchive.com/4–Crossbow∗-datasheet.html.

[30] "TinyOS Documentation," 2013, http://www.tinyos.net.

[31] Y. Zhou, Y. Fang, and Y. Zhang, "Securing wireless sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 3, pp. 6–28, 2008.

[32] S. H. Jokhio, I. A. Jokhio, and A. H. Kemp, "Node capture attack detection and defence in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 2, no. 3, pp. 161–169, 2012.

[33] P. Porambage, P. Kumar, C. Schmitt, A. Gurtov, and M. Ylianttila, "Certificate-based pairwise key establishment protocol for wireless sensor networks," in *Proceedings of 10th IEEE International Conference on Embedded Software and Systems (ICESS '13)*, pp. 667–674, Sydney, Australia, December 2013.

[34] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila, "Two-phase authentication protocol for wireless sensor networks in distributed IoT applications ," in *Proceedings of the IEEE 14th International Conference on Wireless Communications and Networking (WCNC '14)*, pp. 2770–2775, April 2014.